Andy Bartholomew
Honors Thesis
John Donaldson, Advisor

Japanese text segmentation: a comparison of different methods applied to Kanji

Abstract

Written Japanese and Chinese contain no word delimiters such as spaces, so segmentation into words is the first step in processing text in these languages. Over the years several methods of segmentation that utilize various statistical and grammatical principles have been developed. I have implemented two such methods, the Tango algorithm by Ando and Lee and a hidden markov model by Papageorgiou. In this paper, I will contrast these two methods with an open-source knowledge-based parser Juman 5.1 in their ability to segment Japanese text.

Introduction

Written Japanese and Chinese use no spaces and little punctuation to delimit words. Readers instead depend on grammatical cues (i.e. in Japanese, particles and verb endings), the relative frequency of character combinations, and semantic context, in order to determine what words have been written. Segmentation into words is important for programs to efficiently search, summarize, or process text in Japanese and Chinese. Lexical and syntax based segmentation programs have been generally successful, but require very large dictionaries and complex grammars, and are limited to a given language. One example of such a program is the Juman parser[1].

However, there are also methods that employ statistical methods to determine the proper segmentation of a passage. These methods are generally language independent, instead depending on a large body of example text referred to as a training corpus. The hidden Markov

[1] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman.html

model advanced by Papageoriou is one such method, deriving a probabilistic model for the sequence of transitions inside and between words.

There are also language-independent methods that require only some human-supervised training. Ando and Lee's Tango algorithm only requires human-judged parameter tuning and a large unsegmented corpus. The algorithm chooses its word boundaries based on the comparative frequency of sequences that terminate at a possible word boundary and phrases that cross the boundary. However, Ando and Lee only claim good results in segmenting long sequences of *kanji*, which are essentially equivalent to proper nouns, titles, and technical terms with no grammatical information.

The algorithms as put forth by their respective authors have been implemented in order to contrast their performance with each other and with the Juman parser. A large corpus of human-segmented Japanese is needed to train Papageoriou's model. A small collection of human-segmented data has been created to compare with the Juman parser and Tango algorithm, although more data would be beneficial.

Background

Written Japanese uses three writings systems: *hiragana* (ひらがな), *katakana* (カタカナ), and *kanji* (漢字), along with the occasional roman alphabet characters (known as *romaji*). *Hiragana* and *katakana* are phonetic syllabaries, meaning that each symbol corresponds to one syllable rather than a sound as in the roman alphabet. Both these syllabaries contain approximately 46 characters along with some obsolete characters, and are sufficient for writing all of Japanese. In modern written Japanese, *hiragana* is used for grammatical markers called particles, adjective and verb endings, and very common short words. *Katakana* is used primarily

for borrowed words and onomatopoeia, although it can also appear in a variety of other specialized cases.

In contrast, *kanji* are modified Chinese characters, used only for writing nouns and the bases of adjectives and verbs. These characters usually have several pronunciations and meanings. There are 1,945 characters taught in primary and secondary school, with an additional 983 used in personal names. A reader of Japanese must learn most of these characters in order to read a text, although the added meaning of the *kanji* makes reading much easier than if the text were written only in phonetic characters. The presence of *kanji* allows for Japanese and Chinese to be written without word delimiters because they provide an immediate semantic meaning to a given sequence of characters.

The earliest approaches to segmenting Japanese attempted to create structures that represent Japanese grammar. As mentioned above, these programs encode a very large lexicon with grammatical annotations and a defined grammar. According to Papageoriou, one implementation of this method by Mori et al. was able to segment with 95 percent accuracy (Papageorgiou 283). Other lexicon-based approaches start from simpler dictionary-and-grammar based segmentation and then add further verification and disambiguation by various other means (Papageorgiou 283) (Ando and Lee 246).

The most current Japanese parser is Juman 5.1, a free open-source program. Unfortunately, the implementation details are in Japanese, and the only available translation is for version 0.5. Other parsers referenced by the literature are not available online or are not being actively maintained.

Statistical approaches, influenced by similar work on processing speech, trace back to the Yamron et al. in 1993 (Papageorgiou 284). This algorithm calculates all potential segmentations

and chooses the most likely one, using dynamic programming to make this feasible. Teller and Batchelder developed an algorithm using the Hidden Markov Model approach, but directed it specifically towards phrases without *kanji* (284). Ito and Koda also published an algorithm similar to Tango, but their method is directed towards speech recognition (Ando and Lee 146).

Algorithms

Each segmenting algorithm examines the likelihood of a space between each character in a given string. The algorithms are designed to take a string of characters and output an array of Boolean values corresponding to the potential boundary between each character (see figure 1). In figure 1, the output symbols indicate that "ABC" and" XYZ" are distinct words, with a boundary between C and X.

```
A B C X Y Z
 0 0 1 0 0
```

Figure 1

Papageoriou's method uses a hidden Markov model, which views a character sequence as a series of two-character outputs generated by random transitions between states in a finite state machine. A hidden Markov model assumes that only the current state and constant transition probabilities will determine the next state. The Viterbi algorithm uses a hidden Markov model to determine the most likely sequence of hidden states that would result in the given output sequence. It requires a probability for each possible transition and the probability that a given output was generated by each state. The Viterbi algorithm is frequently used in speech recognition and bioinformatics. Pseudocode for the algorithm was referenced in (Jurafsky and Martin 178).

In the case of word segmentation there are two states, word boundary and word continuation, and four transitions. Papageoriou uses a two-character sequence, or bigram, as output so that the hidden state is simply the likelihood that the bigram is at the edge of a word or

inside a word. Given the large number of characters in Japanese, it would seem that the number of bigrams would be extremely large. However, the vast majority of bigrams appear very rarely.

The probabilities for the state model is built by sliding a two-character window over a body of pre-segmented text, counting the relative frequency of each transition and the boundary and continuation states for each bigram. This data is used to calculate the probabilities for each transition and the probability that each bigram is in either state. This training is asymptotically linear with respect to the input, and can be updated with further data at any time. When given a sequence to segment, the program uses the Viterbi algorithm to build the most likely state sequence corresponding to the input. This is asymptotically linear with respect to the input sequence.

Ando and Lee's algorithm is based simply on how frequently characters occur together. Sequences of characters that appear together more frequently are more likely to be words. The algorithm is trained by counting how often each sequence of length two through six occurs in the corpus. To segment a sequence, the algorithm calculates a score for each potential word boundary, choosing word boundaries with scores that are higher than their neighbors or above a determined threshold. The algorithm calculates values by comparing the frequencies of n-grams that terminate at the boundary with those that cross; for each crossing/terminating pair in which the terminating sequence is more frequent, the value is incremented by one. In figure 2, the '?' denotes a potential boundary, with the *s* brackets marking terminating sequences and the *t* brackets marking crossing sequences. In practice, separate values are calculated for each number $2 \leq n \leq 6$ and then averaged for the final value of the boundary. The threshold value used in this implementation was the same as that used by Ando and Lee, namely 0.5.

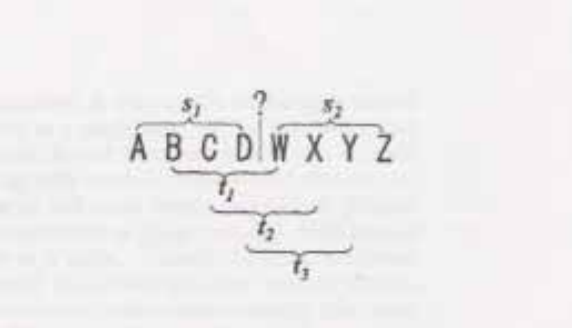

Figure 2 (Ando and Lee 242)

Ando and Lee only claim that this algorithm will work well for long *kanji* sequences. Long *kanji* sequences are usually made up of proper and compound nouns and technical terms, so they frequently contain words that do not appear in the dictionary of a parsing program. Additionally, these sequences contain little grammatical information, as the entire sequence usually serves as a single grammatical entity. Both of these factors make long *kanji* sequences difficult for dictionary and grammar based parsers, and the Tango algorithm was intended to compensate for this. However, the authors specifically avoid claiming that their method works well beyond *kanji* sequences, suggesting that Tango is not likely to perform well on entire sentences of Japanese.

Methods

The collected data for the Tango algorithm consists of a frequency count for every kanji sequence of length 2 through 6. The simplest implementation of this is an associative array between every sequence encountered and the number of times it was encountered in the training data. The trainer locates every kanji sequence of length two or greater in the corpus and add every subsequence to the array. However, this is rather inefficient in terms of training time and memory required.

As an alternative, a prefix tree was implemented with some modifications that accommodate the vast number of possible characters. Each node contains a mapping from possible successive characters to each subtree, and a frequency value for the string corresponding to the path taken to the node from the root. The typical prefix tree is used as a dictionary, with an array of length 26 mapping each letter of the alphabet to a subtree. In this case each node contains a dynamic map rather than a static array to accommodate the widely varying number of successive characters. Every prefix of a string is added to the dictionary at the same time as the string. This method is also much more efficient than directly storing each sequence and its frequency.

Although both methods are asymptotically linear on the size of the training data, the prefix tree is slightly more efficient. For a given sequence of length L≥6 to be incorporated into the training data using the brute-force method, there are L-1 subsequences of length 2, L-2 subsequences of length 3, and so on up to 1 subsequence of length L. If we take the set of subsequences of length 2 through 6, this results in $\sum_{n=2}^{6} n * (L - n + 1)$ additions to our data, which simplifies to $20(L - 1) - 90$ additions. On the other hand, the prefix tree method makes only L-1 additions to the dataset per sequence. Adding a sequence by either method is asymptotically linear with respect to the length of the sequence.

The gain in space between prefix tree and hash table is dependant on the number of distinct sequences occur in a training set, a statistical analysis beyond the current scope of this paper. However, note that every prefix of a path from root to leaf in the tree contains another entry in the data. Each of these entries in a hash table would require another key-value pair.

The training data for the Tango program was taken from the National Institute for Informatics Test Collection for Information Retrieval Systems Project[2], a national Japanese project working on improving computer information retrieval and question answering capabilities in Japanese. The Tango program was trained using a 118 Megabyte set of documents taken from the NACSIS Academic Conference Paper Database (ntc2-j1g), data from the second NTCIR workshop. A set of 1,000 kanji sequences of length 6 or more were extracted from a second corpus (ntc2-j1k) for comparisons. These sequences were segmented by a native speaker of Japanese, the Tango program, and the Juman program. Unfortunately, a large corpus of correctly segmented data is needed to train the hidden Markov model, so results are currently unavailable for that model.

Results

	Human vs. Tango	Human vs Juman	Tango vs Juman
% disagreed phrases	70%	78%	70%
% disagreed slots	27%	23%	24%
disagreements per word	1.2	1.2	1.1
1 bound, 2 not bound	23%	2%	25%
2 bound, 1 not bound	77%	98%	75%

As evidenced in figure #, none of the three segmentations correspond very highly, with approximately one out of every four boundaries in disagreement and a rate of a little more than one disagreement per phrase. The most likely reason for disagreements come from differing granularities of segmentation. The native speaker segmentation was based upon segmenting entire words, but it is possible that Juman and Tango commonly segment prefixes and suffixes as well. This possibility merits further investigation with alternative segmentations by native speakers, particularly in order to more accurately simulate Ando and Lee's metrics.

---

[2] http://research.nii.ac.jp/ntcir/index-en.html

The manner in which Juman disagreed with the native speaker's segmentation, with Juman choosing many more boundaries than the native speaker, merits some discussion. When Juman cannot identify a word, it chooses to segment every character. This happens often with long *kanji* sequences that contain obscure terms and proper names, while a human segmenter can guess the mostly likely segmentation for previously unknown proper nouns.

The Tango segmenter likely had similar difficulty with obscure terms and names, but rather than segment at every character likely chose an incorrect segmentation. Performance could be improved by increasing the size of the training corpus and ensuring that it includes proper names and documents from a wide variety of specialized fields.

Future Work

Obtaining a large corpus of accurately segmented data is the most immediate goal for future work, in order to train Papageorgiou's model. Papageorgiou's work indicates that a corpus containing approximately one million words should be sufficient for segmentation with under 15% error. However, his algorithm was trained and tested using machine segmented data, so further work in establishing the effectiveness of the model in contrast with native speakers would be beneficial.

The exact metrics used by the authors of both methods are somewhat unclear. Further work should be done in order to determine precisely how the authors derived their metrics and whether replicating them is reasonable. It would be worthwhile to compare the metrics used in this research with the metrics advanced by the authors.

Testing of the effectiveness of the Tango method would also benefit from a larger set of experimental data. As mentioned above, Ando and Lee suggest that segmenting methods

perform better at the morpheme level than at the word level, which merits further investigation using data segmented by native speakers at different granularities.

It would also be useful to experimentally measure the relationship between the size of a training corpus and segmentation accuracy for the Tango method. Several other factors in the attributes of a training corpus, such as subject matter, that may play a larger role in the method's performance should be investigated.

Finally, because both statistical methods are language independent, long term investigation should aim at applying the methods to Chinese and any other written language that lacks word delimiters. Ando and Lee applied their algorithm to Chinese and found that it had reasonable results (Ando and Lee 146), but no work has been done applying the hidden Markov model to Chinese.

Considering the very positive results published by the designers of both algorithms and the dearth of recent publications on the topic, it would seem that the problem of segmentation has been solved. Current work in natural language processing is focused on semantics and word disambiguation; that is, training computers to understand the content and meaning of a text. As indicated by the name, the NTCIR is focused on information retrieval.

Regardless of the current focus of natural language processing, statistical models have surpassed linguistic structures as the primary tool. Defining and encoding grammatical models are very complex, but semantic models of natural language appear to be even more challenging. Utilizing statistical tools to understand the meaning of a sentence will likely prove more fruitful. Statistical models are free of language-specific knowledge, so the same methods can be applied to another language without extensive redesign.

Despite the claims regarding segmentation algorithms, readily available web applications perform rather poorly. The most common applications purport to provide dictionary definitions for every word in a passage but frequently make mistakes in segmentation. Before reading the literature, this poor performance gave the impression that the problem of segmentation had not been solved. Perhaps the most useful direction for these programs would involve incorporating them into an online application. Such an application would require fast performance, but would be very useful for students of Japanese or Chinese.

Acknowledgements

I am indebted to the NTCIR for the use of their datasets. I am also thankful to Yuta Sugano for helping to generate segmented data.

Bibliography

Ando, Rie Kubota and Lillian Lee. "Mostly-unsupervised statistical segmentation of Japanese kanji sequences." Natural Language Engineering (2003): 127-149.

—. "Mostly-Unsupervised Statistical Segmentation of Japanese: Applications to Kanji." North American Assocation for Computational Linguistics (NAACL). 2000. 241-248.

Jurafsky, Daniel S and James H Martin. "Speech and Language Processing." New Jersey: Prentice Hall, 2000. 176-179.

Papageorgiou, Constantine P. "Japanese Word Segmentation by Hidden Markov Model." Human Language Technologies Workshop. 1994. 283-288.